MKS

BulletML Pattern Builder

Handbook

by Micah Koleoso Software, 2025
https://micahkoleososoftware.com

Table of Contents

1 Introduction

This document is the Handbook to the MKS BulletML Pattern Builder Software package.

The MKS BulletML Pattern Builder is an editing tool used to create bullet patterns for Shoot ‘Em Up (a. k. a. Schmup) games which can parse and interpret BulletML XML-files as per standard BulletML Specification. This editor supports both the horizontal and vertical bullet pattern development as used in horizontal (left to right axis) and vertical (down to up axis) scrolling shooters.

This tool enables the user to configure and preview bullet patterns within the editor before exporting the data out for further processing. Both the BulletML XML-file and Pattern Builder JSON-files can be used for subsequent parsing and processing in other tools.

The user generated projects are created and stored in the Cloud and can be accessed from any compatible platform such as Linux, Windows and macOS with the Steam Launcher installed and the application valid on the Steam user’s account. To associate the user to their projects created in the application, only the Steam ID of each account is used, no other personally identifiable information is collected or stored in the Cloud.

Each Steam user can be logged in to only one instance of the application at a time regardless of the executing platform and with only the most recent login being active. The successful login requires an application account that is active and not banned or deleted.

BulletML is a quasi-standard definition for bullet patterns in Shoot ‘Em Ups that could originally first be widely encountered in the Arcade machines of the 80s. Later on, these Shooters came to consoles and home computers of the time. Today, these kind of games are still being developed and published.

Knowledge of the BulletML Specification is absolutely essential to make effective us of this application. A link to get this document is given in the References section of this text.

For a Shoot ‘Em Up to provide interesting challenges to the player, various enemies and enemy bosses attack the player with varied and interesting bullet formations that are known as patterns.

The creation of these bullet patterns and making them unique, interesting and challenging for the player has been made easier by the invention of the BulletML Specification as a standard declarative language and file format for designing these sequences of bullets.

The MKS BulletML Pattern Builder aims to provide a visual tool for creating the XML documents that conforms to the defined BulletML standard XML files, so that these files can be used in any games and applications which themselves adhere to the standard. This tool supports both vertical and horizontal oriented pattern development and fully covers the BulletML features defined in the standard, although in this version of the software, actions, fires and bullets are only supported per reference and not inline as sub-objects. No game logic can be implemented in this application and a single bullet pattern, although it can contain multiple hierarchies of bullet emissions, is considered to be a Project and is the sum of assets that will be exported as a BulletML XML-file.

The output from the Pattern Builder tool is either the BulletML XML document that can then be further utilized in other software, as this file conforms to the BulletML file standard, or the project itself can be exported as a JSON file for safekeeping, storage, exchange or further manual modification. The JSON-file does not contain any user identifiable information or data about the creator of the bullet patterns.

The tool's native Pattern Builder JSON-file format contains everything needed to transfer the file between instances of the tool for any supported operating system as long as a Steam Launcher is available there. The text-based file can be mailed, copied or submitted to source control for archiving as required, as it is a document in JSON format. The JSON file project format is documented below and easily human readable which makes it easier for external software to directly parse this file. Manual modification of this file is also possible as long as the structure is retained. Inside the JSON project's file is also all data which forms the contents of an exported BulletML XML file, so the JSON file is complete with all bullet pattern data as well as providing metadata information on the project setup. Furthermore, the JSON file format is versioned and currently available as v1.

The Pattern Builder application is used by creating Projects, and within these Projects a number of Actions, Fires and Bullets that have parameters enabling customization and hierarchical data structures. The preview playing within the editor then shows the results as bullet patterns.

A Application Architecture

The Pattern Builder application consists of multiple software components that provide a front-end client application started from the Steam Launchers, as well as back-end servers that manage and store the user created Pattern Builder Projects in the Cloud. This allows the user access to all their Projects from any platform running Windows 11, various Linux distributions and macOS where the Steam Launcher is available. With the purchase of the Pattern Builder Software, the user can launch the client application as per standard for purchased Steam Apps. Only the most recent instance of the application is considered logged in and can communicate with the backend.

While each user account can only be logged in the Pattern Builder Client application from one platform installation at a time, a move of project work to other platforms and OSes within the same account is seamless, there is no need to transfer anything manually. Terminating the application on one computer and starting it on another platform initiates the Steam login process seamlessly. The last login counts and no further work can be done with the prior application instances running, so shutting down any previous instances of the application is strongly urged.

Project files which are exported to disk locally can be exchanged anytime and imported back in through the Pattern Builder Client, either under the same name or, if the name is modified in the file manually, a new project identical to the source project is created for that user. In both cases, the project becomes available immediately in the Cloud as a user's project after the import process concludes successfully.

Currently, there is a limit of 50 projects actively held in the Cloud for each user's account. Each project can create and manage 100 Actions, 100 Fires and 100 Bullet assets. Exported projects can be archived and swapped in and out unlimited times, so the user's total project number is unlimited even if only 50 projects can be active in the application concurrently.

The application "ticks" at 60 frames per seconds, as this is explicitly defined as the timing for intended BulletML file playback. The application will make a best effort to retain this frame rate but actual runtime playback speed depends on the performance capabilities of the underlying hardware platform and the number and complexity of assets created.

Bullets always emit from the center of the virtual Playfield and they either follow the programmed actions or they can behave relative to the simulated player position. The player position is the small cross in the virtual Playfield and this position can be dynamically altered by moving the mouse in the field during playback. It will result in aimed shots emitting relative to that position if the "Aimed" attribute is set within any asset's direction parameters, when that option is available.

It is currently not possible to simulate more than a single player position in the application and the current BulletML Specification does not support exporting XML-files with multiple players for aimed bullets. If there are additional extensions of the BulletML Specification with more features that would be interesting to use with this editor, please send us of feature requests with appropriate links to documents or sites in that direction, we will be happy to evaluate that for updated software versions in future. You can contact us through our support email address, our Discord server or through the Steam group of our company. Check out the product page at our website for further information.

The path through the Action starts with the first Action defined and this Action is normally called "top" though this is not a hard requirement as a name. Where the logic within the assets branches into depends on the first Action defined and follows through the references to other Actions, Fires and Bullets within that hierarchy. After a full playthrough of a frame, the entire Action hierarchy is repeated if "Play" is selected in the tool. When "Play frame" is clicked in the status bar, a single frame is played and the playback paused afterwards at the end of the execution hierarchy.

In this version of the editor, in-lining Actions, Fires and Bullets is not implemented. However, the same logical tasks can be accomplished using ActionRef, FireRef and BulletRef elements, which themselves reference the Action, Fire and Bullet intended. When exporting the BulletML XML file, the Actions, Fires and Bullets also contain the references between them instead of being in-lined.

Please note that there is currently no Undo functionality implemented. Exporting a Pattern Builder JSON-file at any time during editing in the Editor can save a state quickly because performing a Delete on any Project or Asset removes it permanently from the Cloud without a chance of recovery.

B Project

A project is the encompassing composition of a specific setup and configuration that contains all information relevant to a specific instance of a BulletML XML-document, if this document is to be exported for external consumption later.

Furthermore, there is some meta data required to enable correct playback within the editor, such as the virtual Playfield dimensions and the rank difficulty currently set.

A project can set and adjust the virtual dimensions both horizontally and vertically for the Playfield during project development. This influences the scaling of sprites shown on-screen, their velocity within the virtual Playfield, as well as which axis is considered directly forward natively. Horizontal primary axis is right screen side and vertical primary axis is up to the screen top. As a game, this would differentiate if the bullet patterns are for a horizontal or vertical shoot ‘em up, but these settings are for the editor and playback within the application primarily.

The BulletML attributes contain this setting on export of the XML-file as per BulletML Specification and the interpretation is left to whomever parses the XML in their application or game.

A project can also be exported as a Pattern Builder JSON-file and contains everything needed to re-import this project later or to pass the project to another account. This JSON-file can also be managed with source code archives as expected with it being entirely text-based. The JSON-document is also well defined.

In any case, the project can be recreated in the Pattern Builder application completely from this JSON file.

To duplicate a project, a common workflow is to export a Project’s JSON file, rename the project in the document and the filename, and then re-import this project into the editor. On import, if the project name already exists on the user’s account, a project is updated with the import, otherwise a new project is created and is now in the list of selectable projects.

A user account is limited to maximum 50 projects but any number of projects can be swapped out by exporting and importing the JSON Project files, deleting any exported projects to make space. During the import process, application asset limits as described above are verified before an import is performed.

C Action

An Action is the equivalent of an <action> element as defined in the BulletML Specification. An Action contains a sequence of content elements which make up a logical progression of commands that in turn affect a Bullet if this is spawned during playback of this Action in a frame.

Content in Actions can be commands such as Repeats, FireRefs, ChangeSpeeds, ChangeDirections, Accels, Waits, Vanishes or ActionRefs. The configuration and effect of

theses commands follow their definitions closely as described in the BulletML Specification.

D Fire

A Fire is a specific asset to emit a Bullet, that in turn represents a bullet as visual sprite in the Playfield. Both elements are described in the BulletML Specification. In this editor, the Fire settings are evaluated after the Bullet settings, as such the commands in the Fire parameters take precedent over the Bullet parameters under specific attributes chosen in their respective edit dialogs. Edit Dialogs are part of the user interface as described later on in this text.

A Bullet must be defined to enable a Fire to be edited.

E Bullet

A Bullet is the visual representation of a sprite element in the logic of the sequence of Actions which define a playback frame. A Bullet is spawned by a Fire asset which is in turn triggered by an Action.

A Bullet can be customized visually in the editor with a placeholder sprite, however this information is not part of the BulletML Specification and this information is lost when the BulletML XML-file is exported.

The JSON Project file stores the visual representation selected for playback.

A Bullet can itself trigger Actions. Before playback is possible, the editor takes care to inform the user if circular recursion in any of the active Action references has been accidentally configured. This check is performed for Actions within both Actions and Bullets.

2 Workflows

The playback of a frame runs through all assets which are actively integrated into the logical hierarchy of Actions, Fires and Bullets. The playback starts at the first defined Action in the list. The first Action can't currently be swapped out in the editor but the exported JSON-file can be hand edited to modify the Action list, if required.

An active configuration is the tree of Actions which are called from the first Action, that is always the start of playback, through to all assets that are triggered during the execution of a frame. Any inactive assets are stored with the project but not triggered or parsed.

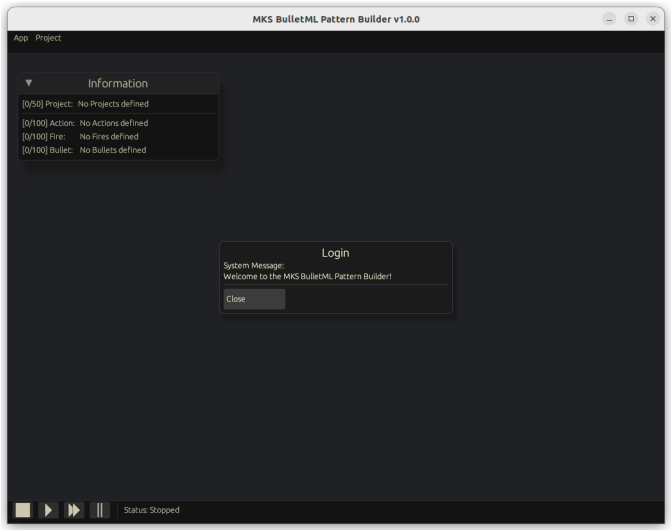
A project can contain up to 100 Actions, 100 Fires and 100 Bullets, even if they are not all called actively during playback. At least 1 Action, Fire and Bullet are needed to have any visual feedback in the editor during playback and these assets must be correctly configured to call each other to successfully see a visual bullet sprite on-screen.

Action, Fires and Bullets can be created anytime. However, to edit and store their modified settings, all required dependent elements of an asset, such as having a Bullet for a Fire asset edit or creating a Fire for use in a FireRef content entry within an Action, must be fulfilled.

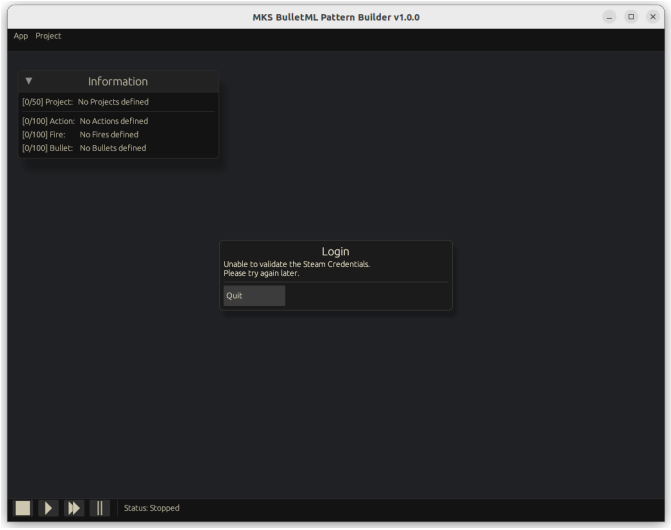
All active assets in a project must have been opened once for editing and stored before the project can be exported, so that all contained elements are valid with default settings applied. Otherwise, the project export will not be possible and the user will be informed the of this fact with a message.

3 Getting Started

As an introduction, the goal here is to show the quick workflow setting up bullet patterns and previewing the results playing in the Playfield of the editor. On starting the application, the main user interface shows the application after a successful login and providing any system information that may be important for working with the tool. In the case shown below, there is no specific message available and the login has been successful.

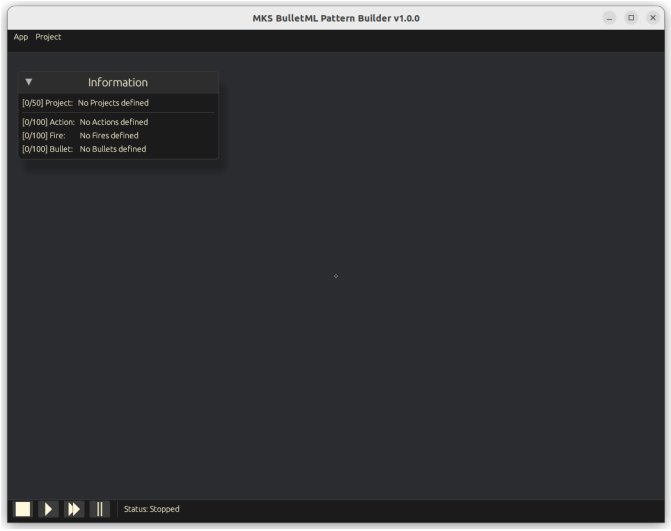


The application is launched through the Steam Launcher and requires a valid purchase on the user's Steam account as well as the execution of the Steam Launcher at all times for validation of the access rights. If the Steam Launcher is not able to verify the correct credentials and purchase or the application account is not in good standing, the Pattern Builder tool will indicate the following on start-up:



Please contact our support per email or come into our Discord Server if you encounter any issues here that we can help with.

After confirmation of the start-up message dialog, the following screen is presented. As we haven't created any projects yet, the menu shows only relevant next steps.



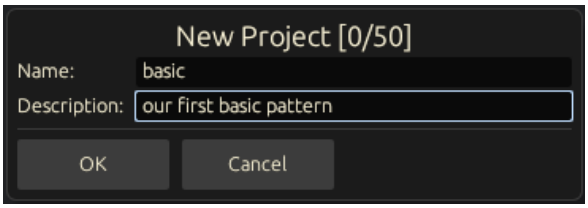
At this stage, we can go through the introductions as described below. Note that the Information Window can be moved per mouse drag in the window title or toggled visible using the F2 key or the sub-menu option “Show/Hide UI Overlays” under the App menu.

A Basic Pattern

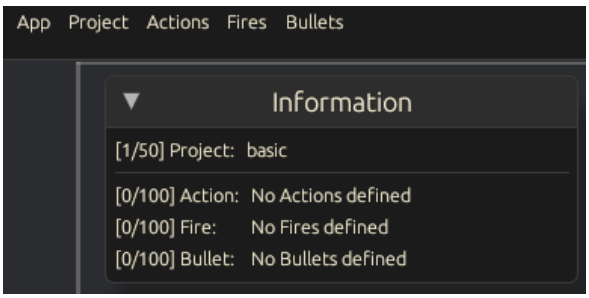
For our basic pattern, let’s start by creating a project. To do this, select the Project menu and click on the “New” menu item. This brings up a dialog where the project name and description can be edited.

Entering a good name and description is strongly urged as the list of projects can get quite large and the data entered here greatly helps differentiate between projects later on.

For this introduction, enter “basic” as project name and “our first basic pattern” as the description.

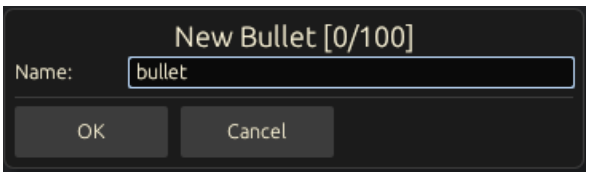


Press “OK” to close the dialog. The menu bar at the top of the application has now expanded fully and will allow us to create the asset content required to see some bullets flying.



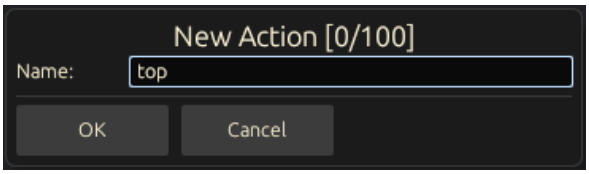
As Actions depend on Fires and Fires depend on Bullets, we will create them in reverse order here.

First, select the “Bullets/New” menu option and we will define a Bullet named “bullet” here as shown below.

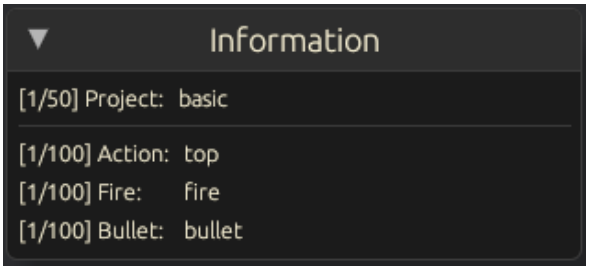


Second, select the Menu “Fires/New” and enter the following data to create a Fire named “fire”. The Fire is used to emit a Bullet when triggered.

Press “OK” here to close the dialog and select the “Actions/New” menu option. As per naming convention, we will name the first Action, which is also where the playback always starts at, “top”.

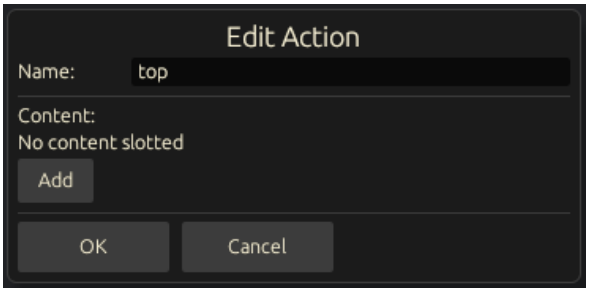


The Information Window now reflects what we have created up until now. We do need to customize the assets though to ensure playback is as we intended.



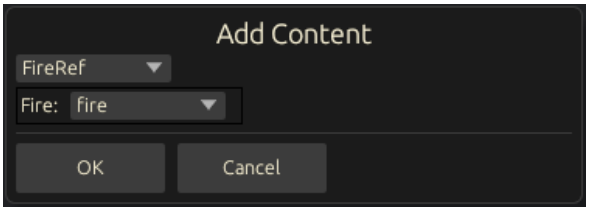
We have used 1 of 50 Projects on our account as stored in the Cloud. In this project, we have created our main Action “top” and a Fire named “fire” to emit a Bullet named “bullet”. We have used up 1 slot of each asset type but could create a further 99 in this project’s budget. These assets shown are also the currently activated elements for any editing we will do. As you switch between multiple assets, the Information window always displays the currently active set of assets.

We can now customize the Action to trigger the Fire in an interval. Select the menu “Actions/Edit”. As the currently active Action asset is “top”, we can now edit it.

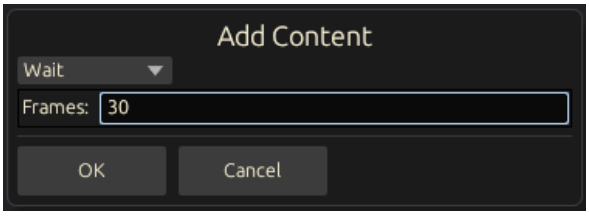


We want to create a Fire emission and then wait 30 frames, which will amount to half a second, until this Action repeats. We will have no further logic in this example.

In the Edit Action dialog, click on “Add” to create a new FireRef content entry from the drop-down widget. As “fire” is the only available Fire asset, it has been preselected.



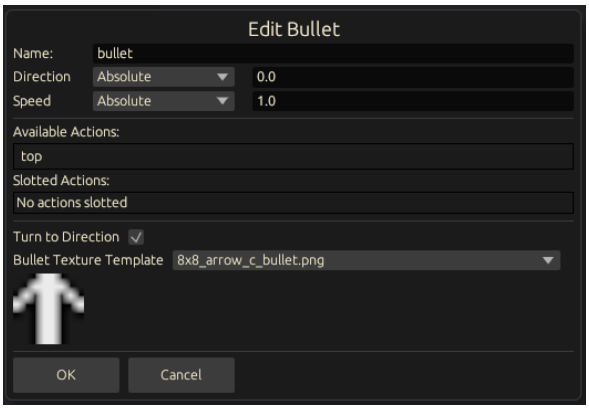
Press “OK” here and see that the “Edit Action” dialog now contains the entry we just created. Click the “Add” button again and create a Wait content entry with a 30 frame wait period.



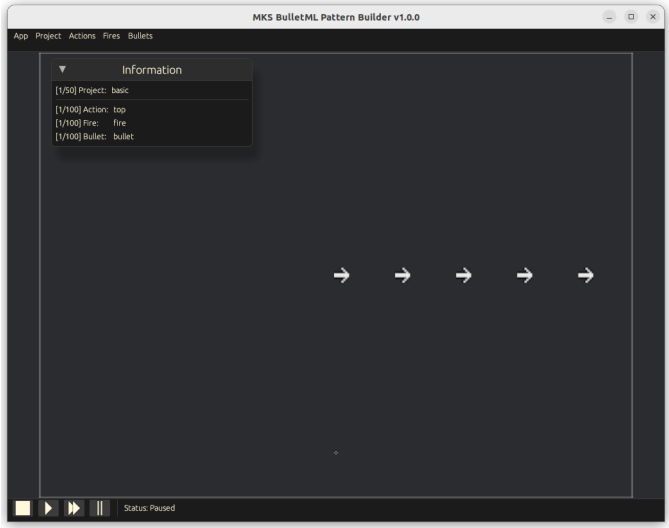
The “Edit Action” dialog should now read as displayed below. We can then move on to the customization of the Fire “fire”.

Press OK here and select “Fires/Edit”. As the currently active Fire is “fire”, we now see the dialog allowing us to customize that specific Fire asset. In this case, we will leave everything as per default and the previously created Bullet “bullet” has already been preset. The direction and speed control the emission velocity of bullets and the parameters customize the behavior of the bullets as they are emitted. Though no parameters are used in this basic example, they can be used to enable parametric inputs to the Bullet velocity at runtime.

To finish this setup, let's edit the Bullet "bullet" by selecting the menu "Bullets/Edit".



Set all options as they are shown above, and make sure to select the Bullet Texture Template option "8x8_arrow_c_bullet.png". Press OK to close this dialog and let's see it all in action.



Press the "Play" button in the status bar of the application. If all went well, you should see a steady stream of bullets being emitted horizontally and leaving the Playfield on the right

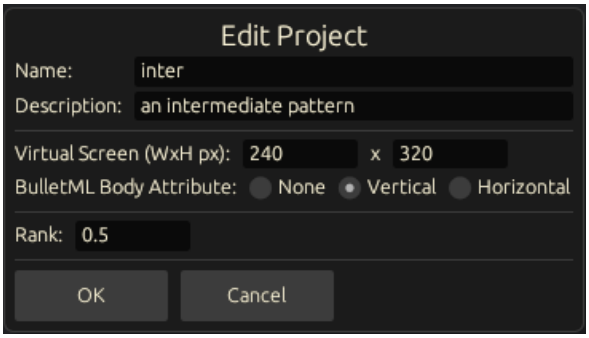
side, as the default setup is configured for a horizontal shooter type game which would scroll towards the right side.

This project has been saved in the Cloud and can be accessed at anytime later and from any application running this Steam account on the list of supported platforms as described in the introduction.

This concludes the first basic introduction to working with this tool. A more involved walk-through follows below.

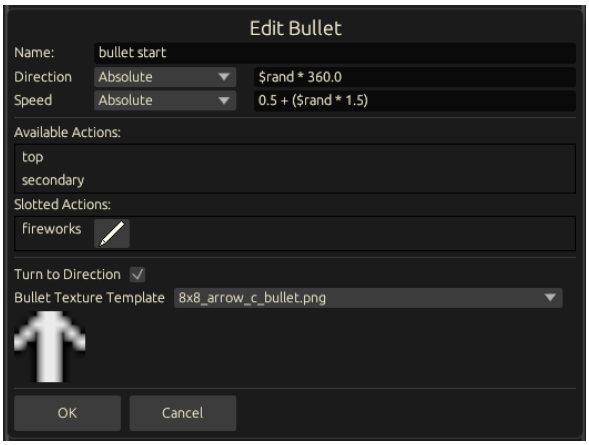
B Randomized Pattern

This pattern is more intermediary and shows the basic interactions between Actions, Fires and Bullets as well how their settings influence each other during playback. How to access the menu for creating and editing the project and assets is described in the basic pattern introduction. To keep the steps simple only the main edits required for this example will be shown as screenshots.



Create a new project named “inter” with description “an intermediate pattern” and set the virtual screen width to “240”, the virtual screen height to “320” and the BulletML Body Attribute to “Vertical”. We want something resembling a vertical shooter play area in a virtual screen resolution that was common in machines in the Arcades of the 80s. Any combination of width and height equal or larger than 64 and smaller or equal to 4096 can be used, but for our purposes, we will use the classic resolution of 240 x 320 virtual pixels. The playback display area and all the sprites displayed within that area are scaled appropriately, regardless of the application actual window size or desktop resolution.

First, create three new Actions named “top”, “fireworks” and “secondary” but don’t edit them at this time. We will customize them later when other dependent assets have been created for them.



Create a new Bullet named “bullet start”. This will be the first bullet we want flying outwards from the screen center. To randomize the direction and speed, we are using an Expression and a variable of BulletML called “$rand” within that Expression. Enter the Expressions as displayed. At runtime, the variable “$rand” generates a value between and including 0.0 and 1.0 and can be used as shown above for dynamics when this Bullet is emitted.

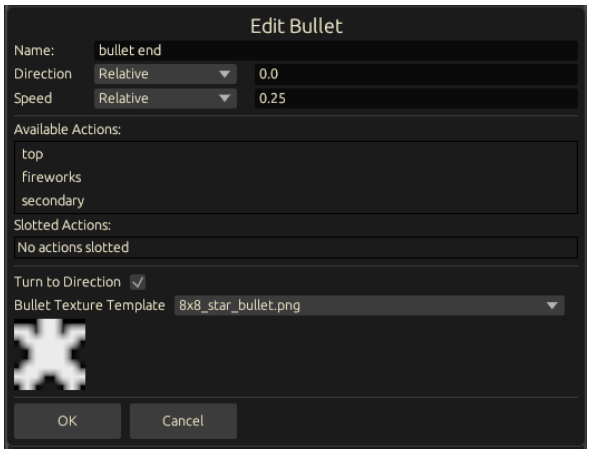
Note that we want to move the Action “fireworks” into the Slotted Actions list by clicking on it in the Available Actions list. Also, set the Arrow as sprite for this bullet to differentiate it visually from another second Bullet we will create later.

Confirm this dialog with “OK” to close it and continue.

Create a new Fire asset named “fire start”. Set the Direction and Speed attributes to “Relative” and make sure “0.0” is set for both. This is to ensure this Fire doesn’t override or influence the Bullet we will emit, as we have custom settings we will enter in the Bullet below.

The “bullet start” Bullet has already been selected per default. Close this too by clicking the “OK” button.

We now edit the Action “top”, which is the first Action we created and will always be the entry point for playback in this project. Add a reference to “fire start” and a Wait of 60 Frames, equivalent to 1 second, here as content for this Action. OK that to leave the dialog.

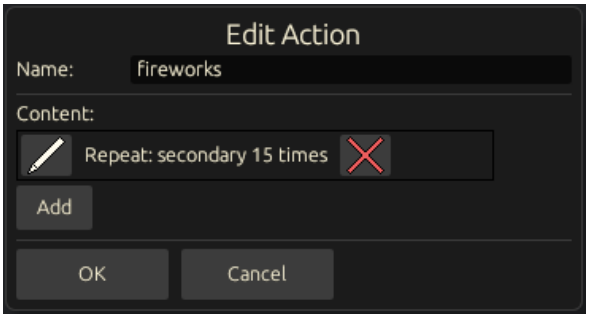


The second Bullet we want to emit is to be created as “bullet end” and contains the parameters as shown above. We have entered Direction and Speed as relative attributes where speed will slightly increase the Bullet velocity when it is spawned, compared to it’s parent Bullet. As this Bullet is a “Child” of the “Parent“ Bullet, it will use that position, direction and speed initially when spawing. The hierarchical elements of the entire BulletML structure is the most creative and complex part of building Bullet patterns and experimentation with these parameters should provide lots of creative and interesting results at playback. For now, we will just use everything as shown above.

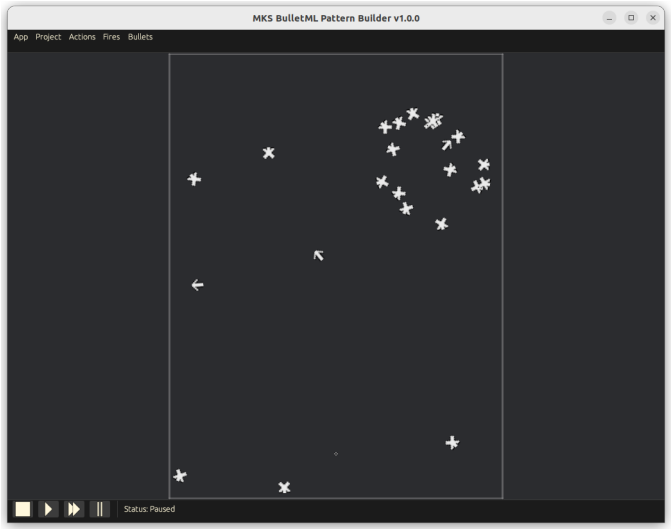
Choose the star sprite as the bullet image to use and leave everything else as is for now. OK out of there to close that dialog.

To emit the second Bullet, create a new Fire named “fire end”. Enter the Expressions as shown above in the appropriate fields and ensure “bullet end” is selected as Bullet to emit. Close this dialog, as we don’t need anything more here.

Edit the Action named “secondary” which we had created earlier and not edited yet at that time. Within it, create a Wait 60 frames asset and reference a Fire named “fire end” as ca FireRef Content here. Close this with “OK” to continue.



As a final part of the setup, edit the Action named “fireworks” to repeat calling the prior Action “secondary” 15 times. This effectively causes 15 Bullets to be spawned through calling the “secondary” Action and each time, a new Bullet of declaration “bullet end” is emitted in a random direction and speed.



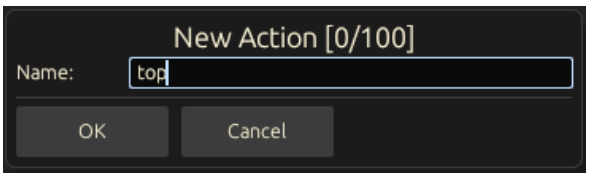
Press “Play” in the status bar of the Pattern Builder application. There should be a fireworks of bullets being emitted. Initially the Arrow bullet generates out of the center in a random direction, and after about a second, each arrow sprite emits a ring of 15 star sprites at the location it is at.

This concludes the second more in depth introduction. There is a lot to experiment with and deep hierarchies can be build where bullets depend on their parent bullet locations etc.

C Aimed Shot

In this introduction, we want to use the player position as a directional goal for the bullets. The player position is demonstrated here as a small cross which appears in the virtual playback area. When the mouse pointer is within the limits of the virtual Playfield, the cross follows the mouse pointer simulating the movement of a player within the field. This can be used to see the effects of logic built with regards to aimed shots at the player.

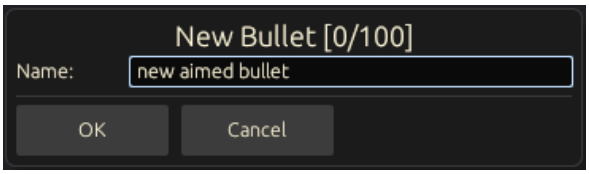
First, create a new project named “aimed” and enter the description text “use the player position for aimed shots”. Press “OK” to leave the dialog and save the settings. From now on, I will assume the user clicks the “OK” button to close the respective dialog as mentioned below. This saves the data entered and continues with the next step.



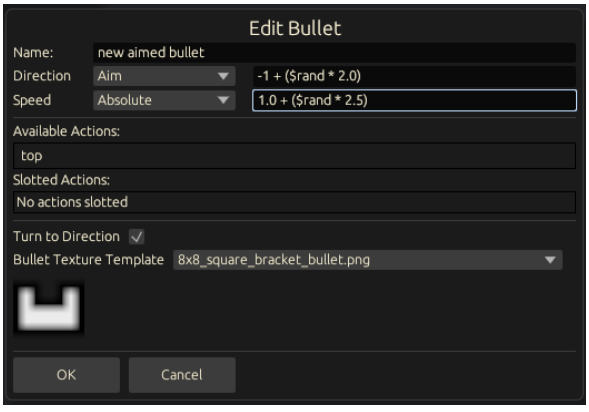
Create a new Action called “top”, which will be our entry point into the playback.

Create a second Action called “spray” but don’t open it for editing yet.

Create a Fire named “aimed fire” and OK the dialog for saving.



As the last step of the creation process, create a new Bullet named “new aimed bullet”. Save that and go on to edit the Bullet we just created.

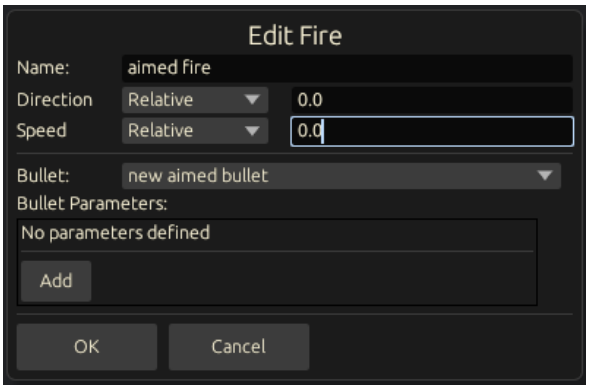


Open the Bullet “new aimed bullet” for editing, it should still be selected as current bullet since it’s the only one and it’s the last one created.

As we want the Bullet to aim at the player position, select the Direction attribute as “Aim”, enter the Expressions as shown above and make sure the Speed attribute is “Absolute”.

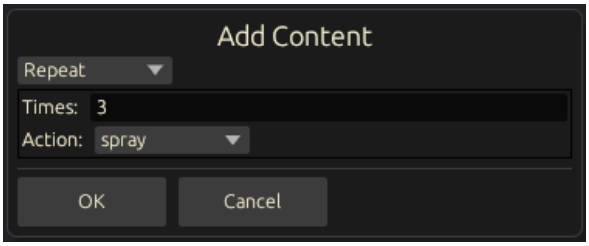
The Direction and Speed settings enable the Bullet to be emitted having the direction to the player with up to 1 degree random variance angle to each side. The Speed Expression enables the bullet to have a movement velocity of between 1.0 and 2.5 pixels total per frame. At a playback rate of 60 frames per second, bullets will really be flying!

For this example, I chose the square bracket sprite as the image. Leave all other settings as per default.

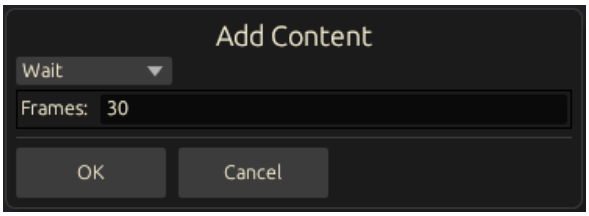


Next, open the Fire “aimed fire” for editing. We set Direction and Speed to “Relative” and the values to “0.0” to prevent this Fire having any further influence over the Bullet “new aimed bullet” it will emit. Both Fires and Bullets can have attributes set, and the Bullet is evaluated first, then the Fire. So it is possible, that the Fire can completely override the Bullet values at runtime, this depends which attributes are set for Direction and Speed in both edits.

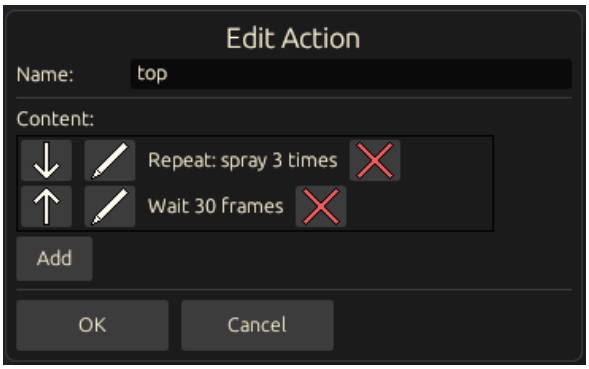
Select and open the “spray” Action to enter a FireRef content with the Fire “aimed fire” to be referenced. Save that and continue. For the next two steps, open the “top” Action for editing.



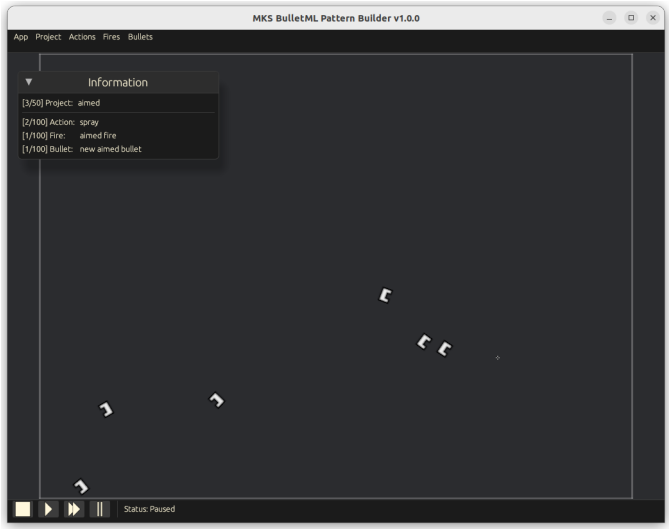
In the “top” Action, add content for a repeat with 3 times set and let it call the “spray” Action each time.



Add another content to the “top” Action, add a Wait with 30 frames, which should be half a second.



The final contents of the Action “top” are shown above. We are calling the “spray” Action 3 times and then waiting half a second before this is looped again.



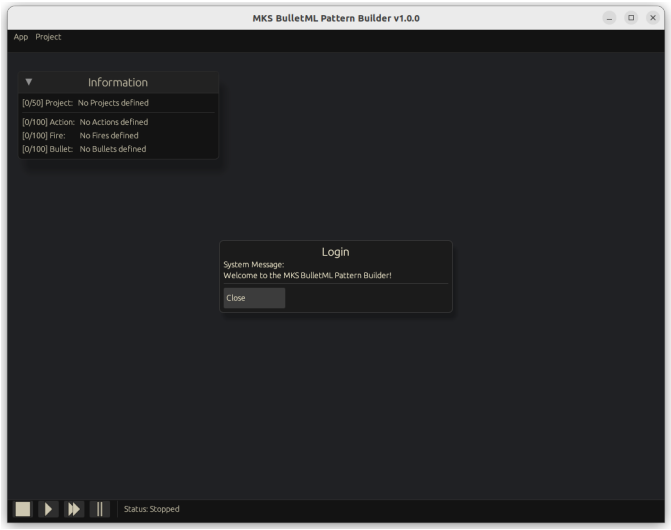
After pressing the “Play” button in the status bar, you can see bullets being emitted around the position of the player position, shown as a cross, in the virtual Playfield. In this case, we are emitting 3 bullets from the center of the Playfield to an angle of -1 to +1 degrees around the player position.

When the mouse cursor leaves the Playfield, a standard player position in the center bottom half of the field is fixed as player position representative, regardless of the type of Playfield BulletML type attribute (None, Horizontal, Vertical) set. Move the mouse in the Playfield to simulate the movements of a player craft.

4 User Interface

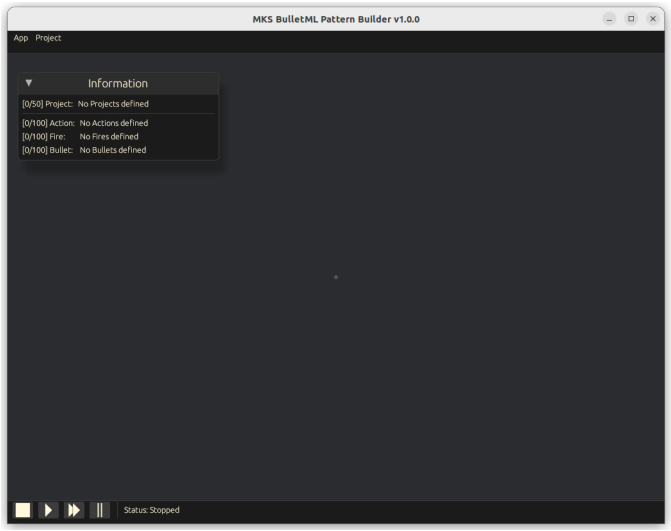
Following is the display and description for each user interface element with the application.

A Start Screen



On successful login, the System Message is presented. Important information, such as maintenance down-times or upcoming version changes will be communicated here at application startup.

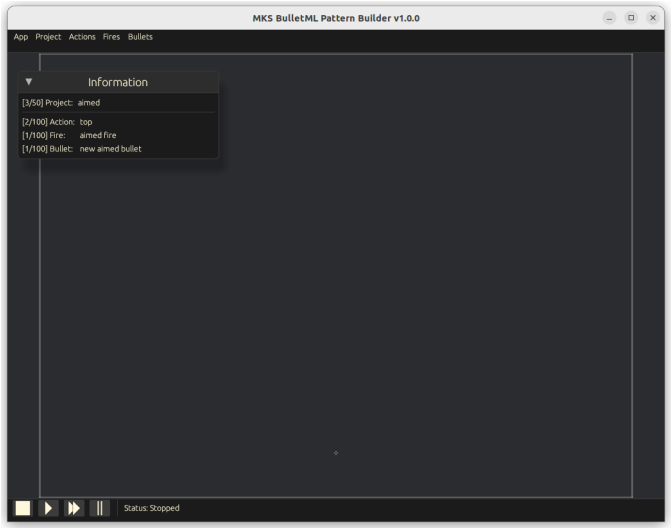
1 Start Screen without Projects available



Every new, validated user begins with an empty list of projects. The projects will be stored in the Cloud as they are created in time.

Creating a project will activate additional menu bar options for further defining assets in the currently active project.

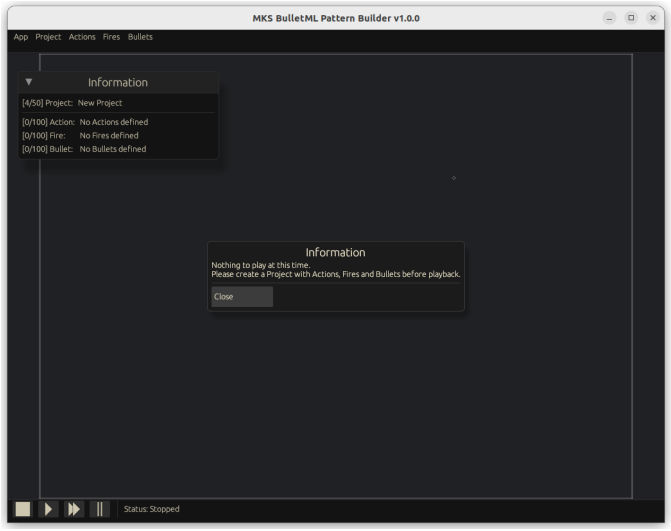
2 Start Screen with Projects available



The user having created at least one project in a prior editing session will restart with the last active project loaded and the information window displaying the last selected Action, Fire and Bullet, if they exist.

The Information window shows the total number and maximum count limit of each type of asset that has been created in the project.

3 Nothing To Play



If a project has been created but not any assets within that would enable playback of the Action hierarchy, the user is informed of this fact. At least 1 Action, 1 Fire and 1 Bullet are needed to show some visual sprite during playback. However, the assets must be configured to use these elements too.

In the Information window, this clearly shows when no assets have been defined.

See the introductions for an easy example to get something visual up-and-running.

B Menu Bar

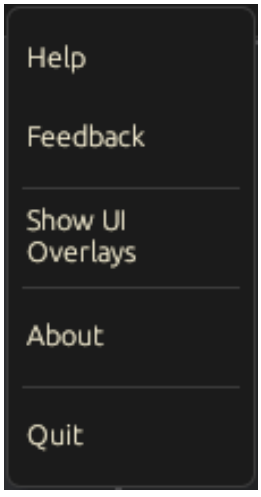
The application provides five total menu drop-downs in the menu bar at the top of the window.

The App menu contains all dialogs which are relevant to the application as a whole.

The Project menu is for the handling of projects and determining which project is to be current for working on. The export of the project as a Pattern Builder JSON-file and a BulletML XML-file can be initated here. Otherwise, changes in projects are automatically stored in the Cloud and do not explicitly need to be saved.

The three remaining menus, Actions, Fires and Bullets, handle the assets which make up the playback for a frame and based on how they are chained together, produce a unique and interesting visual pattern of bullets during play.

1 App Menu



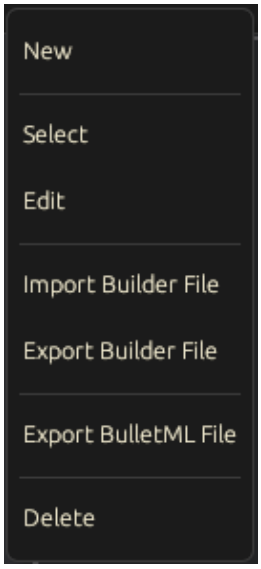
Help – Show the Help dialog with keyboard shortcuts.

Feedback – Shows a dialog which allows the user to send Feedback to us. General feedback, bug reports and feature requests can be formulated, please select the most relevant category for your message. We are happy to get your feedback using our Editor and will evaluate every message we receive.

Show/Hide UI Overlays – Toggle the Information window visibility. The window can also be folded to just its' title bar or hidden completely.

About – Shows the applications splash screen and application version information.

2 Project Menu



New – Opens a dialog to create a new project. A maximum of 50 projects can be active in the user's account.

Select – Shows a dialog which lists all the user's projects.

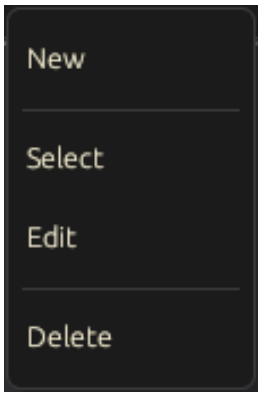
Edit – Open the currently selected project for editing project-wide settings.

Import Builder File – Allows the user to select a file and folder from which a Pattern Builder JSON-file should be imported. The user will see an appropriate message depending on the import results. Note that asset limits apply to and are checked for imported projects.

Export Builder File – Allows the user to select a file and folder into which a Pattern Builder JSON-file is exported. This file contains everything needed to transfer the project outside of the editor or reimport it to another editor instance. For projects in the same user account, all projects are available on all platforms the editor is executed on, as they are stored in the Cloud.

Delete – Opens a dialog which enables the user to delete the current active project. Note that this is permanent and can't be undone.

3 Actions Menu



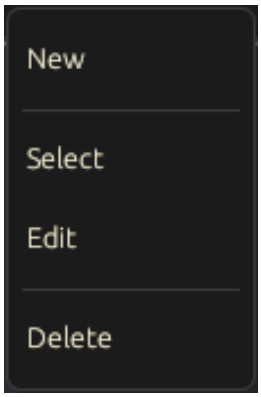
New – Opens a dialog to create a new Action asset.

Select – Opens a dialog to select the Action to be made current.

Edit – Opens a dialog to edit the details of the currently active Action.

Delete – Open a dialog showing the currently active Action which would be deleted if the action is confirmed. The first available Action is then selected as active if the prior Action deletion is confirmed. Note that this is permanent and can’t be undone.

4 Fires Menu



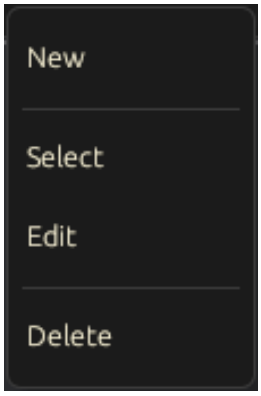
New – Opens a dialog to create a new Fire asset.

Select – Opens a dialog to select the Fire to be made current.

Edit – Opens a dialog to edit the details of the currently active Fire.

Delete – Open a dialog showing the currently active Fire which would be deleted if the action is confirmed. The first available Fire is then selected as active if the prior Fire deletion is confirmed. Note that this is permanent and can’t be undone.

5 Bullets Menu



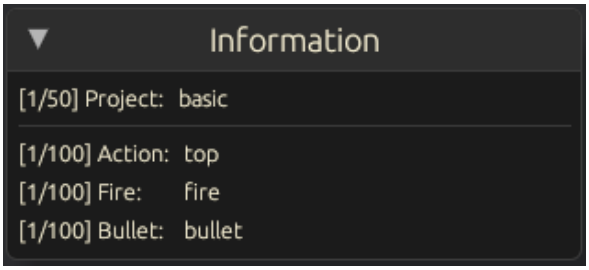
New – Opens a dialog to create a new Bullet asset.

Select – Opens a dialog to select the Bullet to be made current.

Edit – Opens a dialog to edit the details of the currently active Bullet.

Delete – Open a dialog showing the currently active Bullet which would be deleted if the action is confirmed. The first available Bullet is then selected as active if the prior Bullet deletion is confirmed. Note that this is permanent and can't be undone.

6 Information Window



The Information window shows the current number of projects in the user's account as well as the name of the project which has been selected as active to work on.

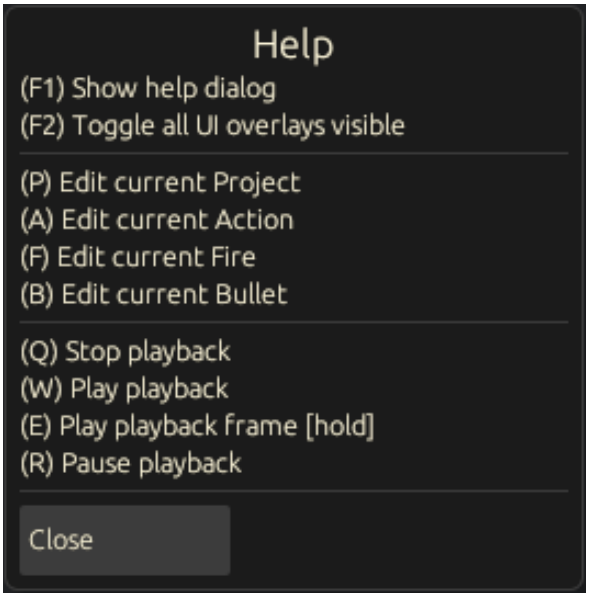
The same goes for the Action, Fire and Bullet number and names. These are the assets which belong to the currently active Project.

There is a limit of 50 for Projects and 100 for all assets within a project.

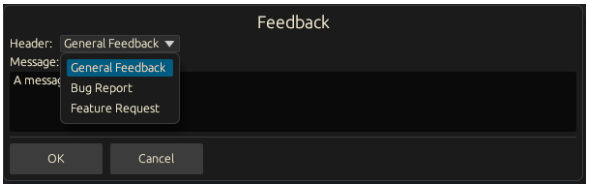
This window can be moved and folded to just its' title bar, or hidden completely to have a clear Playfield.

C Dialogs

1 App Dialogs



The Help dialog shows which keyboard shortcuts can be used to trigger specific common actions while working with the Pattern Builder editor.



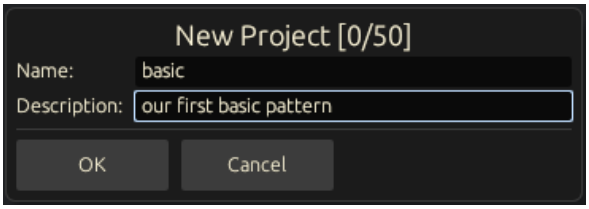
The Feedback dialog enables us to collect your messages to help us improve our product, recognize problems you may be having working with this tool or see what you would be interested in as new features for updated versions of the application.

Please use the right category for your message to speed up our evaluation and response to your feedback here.

The About dialog shows the application splash screen but also provides a quick way to get to our website by clicking on the button shown with the link there.

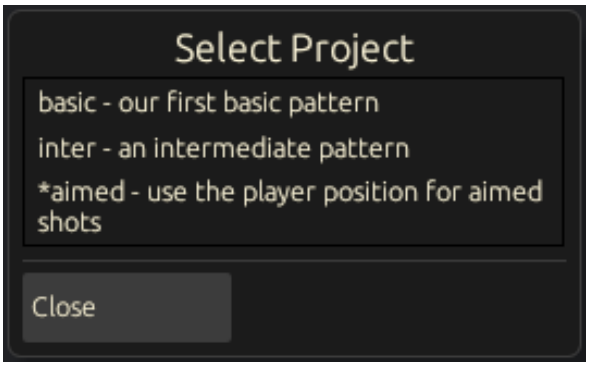
The Quit dialog ends the work session with the Pattern Builder and shuts down the application, if this is confirmed by clicking the “OK” button. All project information and assets will have been stored, so no data can be lost when the tool is shutdown.

2 Project Dialogs

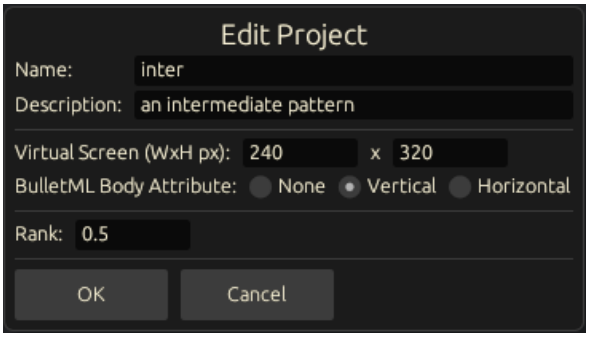


For creating a new project, enter a unique name here, otherwise the creation can't be confirmed. Entering a good description too will show the purpose of the specific project more clearly and help with orientation as the list of projects grows.

The dialog show the number of projects already created as well as the limit as to how many projects can be created in total.



This dialog shows the list of existing project names and descriptions as given. The Project that is currently active is shown with a star (*) in front of the name. Clicking on any other project in the list will select that project as active.

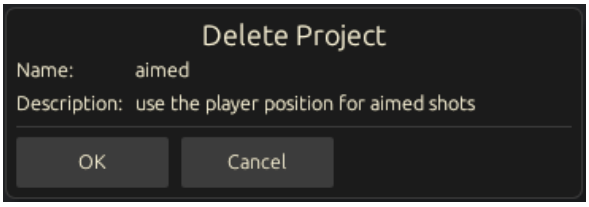


This dialog enables the user to setup the main parameters which will be valid for the entire project. The name and description can be edited here but the name must remain unique to allow the changes to be confirmed with "OK".

The virtual Screen Width can be any geometry combination of 64 by 64 pixels up to 4096 by 4069 pixels for width and height. The Playfield display in the main window area will scale the placeholder sprites and screen ratio appropriately.

The BulletML Body Attribute is a setting used for determining where the main axis is for the project. “None” and “Horizontal” assume the right side of the application’s window is the main axis, “Vertical” assumes this to be the top of the window. This parameter is also exported to the BulletML XML-file as part of <bulletml> type specification.

The rank is a dynamic value which can be set to influence the playback of a frame during editing, as this value can be used as a parameter in the Expressions defined in Actions, Fires and Bullets.

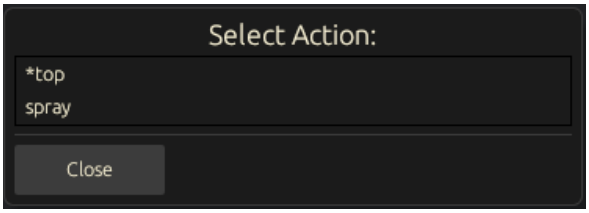


This dialog shows the name and description of the currently active Project which would be deleted if the action is confirmed. The first available Project is then selected as active if the prior Project deletion is confirmed. Note that this is permanent and can’t be undone.

3 Actions Dialogs

For creating a new Action enter a unique name here, otherwise the creation can’t be confirmed.

The dialog show the number of Actions already created as well as the limit as to how many Actions can be created in total.

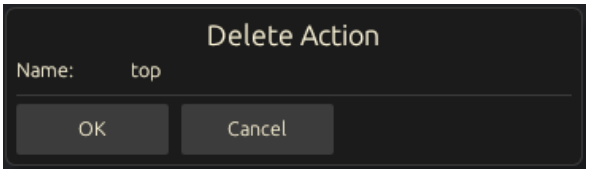


This dialog shows a list of Actions that have been created by the user. The currently active Action is marked with a Star (*). Clicking on any list entry without the Star selects this Action as active for editing and deletion. Clicking the “Close” button closes the dialog without changing anything.

The Edit Action dialog collects all content entries which make up the functionality of the active Action during playback.

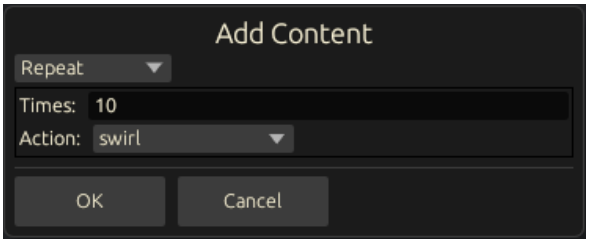
The name of the Action can be edited here but must remain unique to all existing Actions in the current Project.

Content elements can be added, edited, reordered and deleted in the Content list as required, and multiple content entries of the same type can occur. The elements which make up the Content list are parsed top to bottom at playback during each frame. The types of content which can be added into this list are described below.

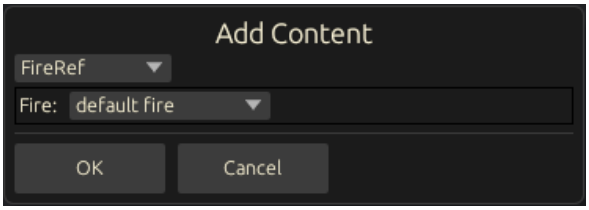


The Delete dialog permanently removes the currently active Action shown by its' name. If this action is confirmed, the asset is permanently removed and can't be recovered. A new asset with the same name can be created again if required.

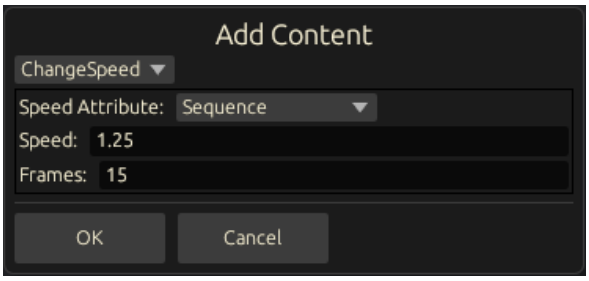
Action Add Content Dialogs



This dialog adds a Repeat Content element. The selected Action is triggered by Times amount successively.

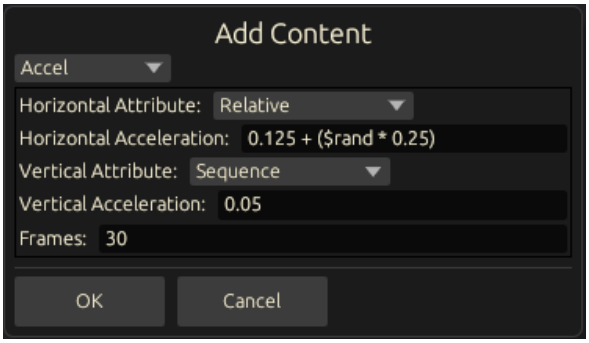


The Fire referenced here is triggered in this Content element.



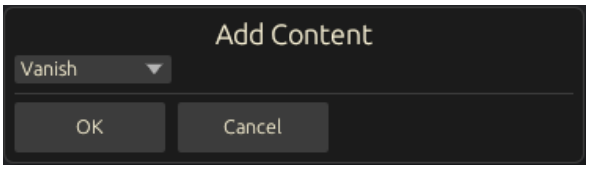
A bullet change of speed can be triggered here. There are multiple Speed Attribute settings available and the Expressions Speed and Frames determine how these parameters are evaluated.

A bullet change of direction can be triggered here. There are multiple Direction Type settings available and the Expressions Direction Degrees and Frames determine how these parameters are evaluated.

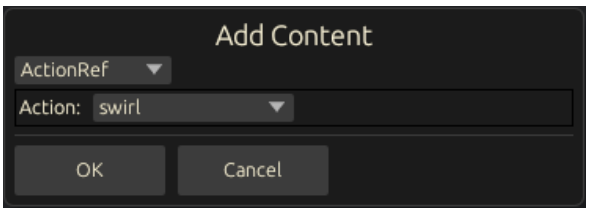


An Accel Content entry determines additional two dimensional forces acting on the velocity of a bullet for the duration of the Frames given. These forces act in addition to other parameter effects which may be in effect due to prior configures types of content in the parent Action list.

The Wait Content effectively blocks further parsing of Content elements in the Action's list further down for a specific number of frames. 60 frames count as 1 second. Existing effects which are active on the relevant Bullet are still processed during the waiting time.



A Vanish Content entry immediately removes the currently referenced bullet from the Playfield. Without a Vanish the bullet is tracked until it leaves the bounds of the Playfield, at which time it is deleted.



The Action referenced here is triggered in this Content element. This feature enables hierarchical triggering of Actions which in turn enables a lot of interesting effects with the bullet patterns. Both Actions and Bullets can contain Actions themselves and their effects can influence each other depending on the settings chosen while editing each asset.

4 Fires Dialogs

For creating a new Fire enter a unique name here, otherwise the creation can't be confirmed.

The dialog show the number of Fires already created as well as the limit as to how many Fires can be created in total.

This dialog shows a list of Fires that have been created by the user. The currently active Fire is marked with a Star (*). Clicking on any list entry without the Star selects this Fire as active for editing and deletion. Clicking the "Close" button closes the dialog without changing anything.

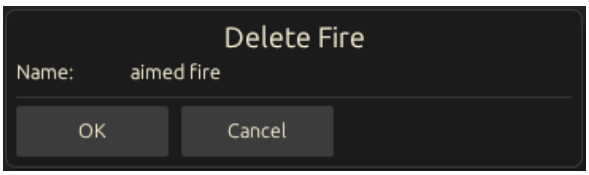
The Edit Fire dialog allows the properties of a newly emitted Bullet to be set. All parameters implement the BulletML Specification.

The name of the Fire can be edited here but must remain unique to all existing Fires in the current Project.

Direction and Speed determine the velocity and angle of the bullet. Depending on the settings here, the initial Bullet parameters may be overridden or adjusted by the Fire values.

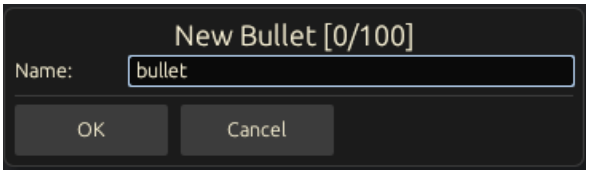
The visual representation of the bullet can be selected from the list given in the drop-down. This sprite is used only for differentiation in the Playfield during playback and is not

exported to the BulletML XML-file. Bullet parametera enable the parametric configuration of the Bullet for variation instead of requiring multiple similar bullets to be built with differentiated behavior. Parameters can contain Expressions.



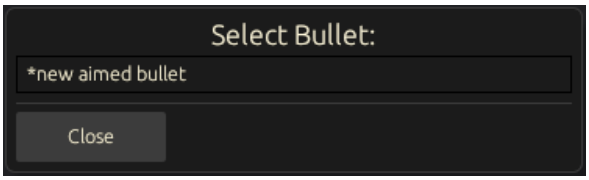
The Delete dialog permanently removes the currently active Fire shown by its' name. If this action is confirmed, the asset is permanently removed and can't be recovered. A new asset with the same name can be created again if required.

5 Bullets Dialogs

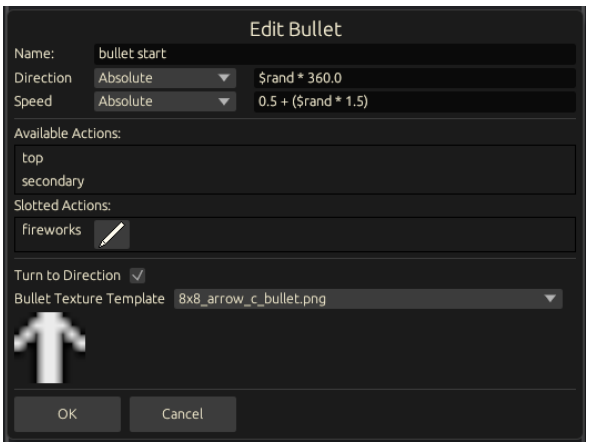


For creating a new Bullet enter a unique name here, otherwise the creation can't be confirmed.

The dialog show the number of Bullets already created as well as the limit as to how many Bullets can be created in total.



This dialog shows a list of Bullets that have been created by the user. The currently active Bullet is marked with a Star (*). Clicking on any list entry without the Star selects this Bullet as active for editing and deletion. Clicking the "Close" button closes the dialog without changing anything.



The Edit Bullet dialog allows the properties of a Bullet to be set. All parameters implement the BulletML Specification.

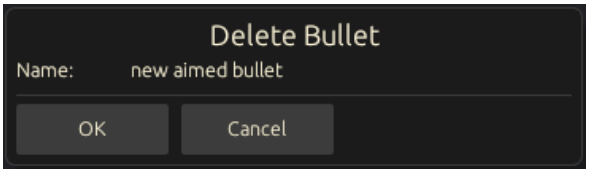
The name of the Bullet can be edited here but must remain unique to all existing Bullets in the current Project.

Direction and Speed determine the initial velocity and angle of the bullet.

A Bullet can also trigger Actions. The available Actions can be moved into the Slotted Actions list to activate them and additional parameters can then be set for each slotted Action. If circular References of Actions are created in error here, the Pattern Builder will warn of this when a playback is attempted.

Clicking Actions in the Slotted List removes them back into the Available Actions list.

The Bullet can set a visual representation used by the Editor in the Playfield for differentiation during playback. Set the “Turn to Direction” flag to orient the Sprite in the direction of it’s velocity, otherwise the sprite is rendered facing to the application window top. This visual representation is not exported as part of the BulletML XML-file.



The Delete dialog permanently removes the currently active Bullet shown by its’ name. If this action is confirmed, the asset is permanently removed and can’t be recovered. A new asset with the same name can be created again if required.

6 Status Bar

The status bar contains the elements used to control the frame playback of the currently active assets as per configuration setup using Actions, Fires and Bullets. The buttons perform as known from other media playback controls, so from left to right, you get Stop, Play, Play frame and Pause by clicking on these elements or using the short cut keys described in the Help dialog.

Using the keyboard key for playing a frame differs from the mouse click on the button by allowing the user to play back multiple frames as long as the key is pressed and held. A mouse click always plays a single frame only.

The status text shows the current state of playback. A single click to play back a frame may be so quick that no status message appears to show, but stopped, playing and paused states can be seen clearly as the frames are executed continuously.

5 Expressions

Any valid mathematical expression used in the appropriate parameter field in the dialogs can contain additions, subtractions, multiplications, divisions, modulo and brackets.

Variables as $1, $2, $3...$10 for parameters can be configured and entered as param sets where allowed in ActionRef, FireRef and BulletRef dialogs. A maximum of 10 parameters containing any valid Expression can be defined per reference parameter block given.

$rand can be used for generating a random value between from 0.0 to 1.0, both inclusive.

$rank is placeholder for the game difficulty between 0.0 to 1.0, also inclusive. This is set in the project edit dialog and is global for the entire project, but can be changed during development in the project editor dialog to see the effect on patterns during subsequent playback.

If the Expression does not evaluate to a value, the dialog containing the Expression may not allow closing with saving of the changes. This can be seen by the hiding of the “OK” button in that case.

6 File Formats

A BulletML – XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bulletml PUBLIC "-//Nakamura//DTD BulletML 1.0//EN"
"http://www.dan-ball.jp/bulletml/bulletml.dtd">
<bulletml type="horizontal" xmlns="http://www.asahi-net.or.jp/~cs8k-cyu/bulletml">
  <action label="top">
    <repeat>
      <times>3</times>
      <actionRef label="spray">
      </actionRef>
    </repeat>
    <wait>30</wait>
  </action>
  <action label="spray">
    <fireRef label="aimed fire">
    </fireRef>
  </action>
  <fire label="aimed fire">
    <direction type="relative">0.0</direction>
    <speed type="relative">0.0</speed>
    <bulletRef label="new aimed bullet">
    </bulletRef>
  </fire>
  <bullet label="new aimed bullet">
    <direction type="aim">-10.0 + ($rand * 20.0)</direction>
    <speed type="absolute">1.0 + ($rand * 2.5)</speed>
  </bullet>
</bulletml>
```

Exporting a file in the BulletML XML-file format gives you a BulletML XML-file which can be validated with external Tools and that contains all elements which make up a bullet pattern. All files emitted by the Editor contain standard XML and BulletML header information and at least one <bulletml> section which lists all <action>, <fire> and <bullet> sub-elements as they were created in the tool.

The parameters within the elements are handled as per standard BulletML specification and all possible values can be found in the file if these are used in the editor to configure them appropriately. Note that no inline <action>, <fire> or <bullet> elements are emitted by the Editor at this time but rather are reference-linked.

External games and tools which conform to the parsing of BulletML XML-files should be able to play back these files as exported.

B Pattern Builder Project – JSON


```
{
  "exporter": "MKS BulletML Pattern Builder",
  "file_version": 1,
  "mks_website": "https://www.micahkoleososoftware.com",
  "name": "aimed",
  "creation_date": "2025-08-10 19:44:00",
  "description": "use the player position for aimed shots",
  "virtual_screen_dims": [
    320.0,
    240.0
  ],
  "difficulty": 0.5,
  "bulletml": {
    "attribute": "HORIZONTAL",
    "contents": {
      "actions": [
        {
          "attribute": {
            "text": "top"
          },
          "contents": {
            "action_contents": [
              {
                "Repeat": {
                  "contents": {
                    "times": {
                      "contents": {
                        "term": {
                          "text": "3"
                        }
                      }
                    },
                    "action_ref": {
                      "attribute": {
                        "text": "spray"
                      },
                      "contents": {
                        "params": []
                      }
                    }
                  }
                }
              },
              {
                "Wait": {
                  "contents": {
                    "frames": {
                      "text": "30"
                    }
                  }
                }
              }
            ]
          }
        },
        {
          "attribute": {
            "text": "spray"
          },
          "contents": {
            "action_contents": [
              {
                "FireRef": {
                  "attribute": {
                    "text": "aimed fire"
```

```
            },
            "contents": {
              "params": []
            }
          }
        }
      ]
    }
  }
],
"fires": [
  {
    "attribute": {
      "text": "aimed fire"
    },
    "contents": {
      "direction": {
        "attribute": "RELATIVE",
        "contents": {
          "degrees": {
            "text": "0.0"
          }
        }
      },
      "speed": {
        "attribute": "RELATIVE",
        "contents": {
          "speed": {
            "text": "0.0"
          }
        }
      },
      "bullet_ref": {
        "attribute": {
          "text": "new aimed bullet"
        },
        "contents": {
          "params": []
        }
      }
    }
  }
],
"bullets": [
  {
    "attribute": {
      "text": "new aimed bullet"
    },
    "contents": {
      "direction": {
        "attribute": "AIM",
        "contents": {
          "degrees": {
            "text": "-10.0 + ($rand * 20.0)"
          }
        }
      },
      "speed": {
        "attribute": "ABSOLUTE",
        "contents": {
          "speed": {
            "text": "1.0 + ($rand * 2.5)"
          }
        }
      },
      "action_refs": []
    },
    "turn_to_direction": true,
    "bullet_texture_name": "8x8_square_bracket_bullet.png"
```

```
        }
      ]
    }
  },
  "current_action_index": 0,
  "current_fire_index": 0,
  "current_bullet_index": 0
}
```

The Pattern Builder JSON-file output contains the entire project consisting of project-specific settings as well as all elements that make up the sequence of Actions, Fires and Bullets for playback and export as BulletML XML-files.
These files reflect all meta data settings of the project which is used in the Editor for playing a frame, such as Playfield dimensions, the main axis vertically or horizontally, which difficulty was set and which assets were selected last.
In addition to that, all created assets with their hierarchy, parameters and Expressions can be found within this file.
For visual representation of a Bullet during playback in the Editor, a sprite texture can be selected from a list of available images, shown here as part of the “bullet” element definition:

"bullet_texture_name": "8x8_square_bracket_bullet.png".

Note that this is not exported as part of a BulletML XML-file, as this is not part of the BulletML Specification and interpretation of the XML-file is left to the parser of the file.
The format of the JSON-file will be versioned and can be relied on as a source for further processing in a subsequent workflow. At this time, the file version is v1.

7 References

BulletML Format Definition - https://www.asahi-net.or.jp/~cs8k-cyu/bulletml/index_e.html

MKS BulletML Pattern Builder Homepage – TBD

MKS BulletML Pattern Builder Steam Page - TBD

15th August, 2025	v1.0 - Initial version.